



Search a new era

David Pilato

 @dadoonet

 @pilato.fr

Normandie.ai

Les Journées Normandes des intelligences artificielles responsables

Elasticsearch

You Know, for Search



Elasticsearch

APACHE
LUCENE™



66

These are not the droids
you are looking for.

```
GET /_analyze
{
  "char_filter": [ "html_strip" ],
  "tokenizer": "standard",
  "filter": [ "lowercase", "stop", "snowball" ],
  "text": "These are <em>not</em> the droids
          you are looking for."
}
```

```
"char_filter": "html_strip"
```

These are **not** the droids you are looking for.



These are not the droids you are looking for.

```
"tokenizer": "standard"
```

These are not the droids you are looking for.



These
are
not
the
droids
you
are
looking
for

"filter": "lowercase"

These	→	these
are		are
not		not
the		the
droids		droids
you		you
are		are
looking		looking
for		for

"filter": "stop"

These	→	these	→	
are		are		
not		not		
the		the		
droids	→	droids	→	droids
you		you		you
are		are		
looking		looking		looking
for		for		

"filter": "snowball"

These	→	these	→	droids	→	droid
are		are		you		you
not		not				
the		the				
droids	→	droids	→	droids	→	droid
you		you		you		you
are		are				
looking		looking		look		look
for		for				

These are `<em>not</em>` the **droids** **you** are **looking** for.

```
{ "tokens": [{  
  "token": "droid",  
  "start_offset": 27, "end_offset": 33,  
  "type": "<ALPHANUM>", "position": 4  
}, {  
  "token": "you",  
  "start_offset": 34, "end_offset": 37,  
  "type": "<ALPHANUM>", "position": 5  
}, {  
  "token": "look",  
  "start_offset": 42, "end_offset": 49,  
  "type": "<ALPHANUM>", "position": 7  
}]}
```

Semantic
search
≠
Literal
matches

similarweb

**YOU'RE COMPARING
APPLES TO NECTARINES**



Elasticsearch

You Know, for **Vector** Search

What is a
Vector?



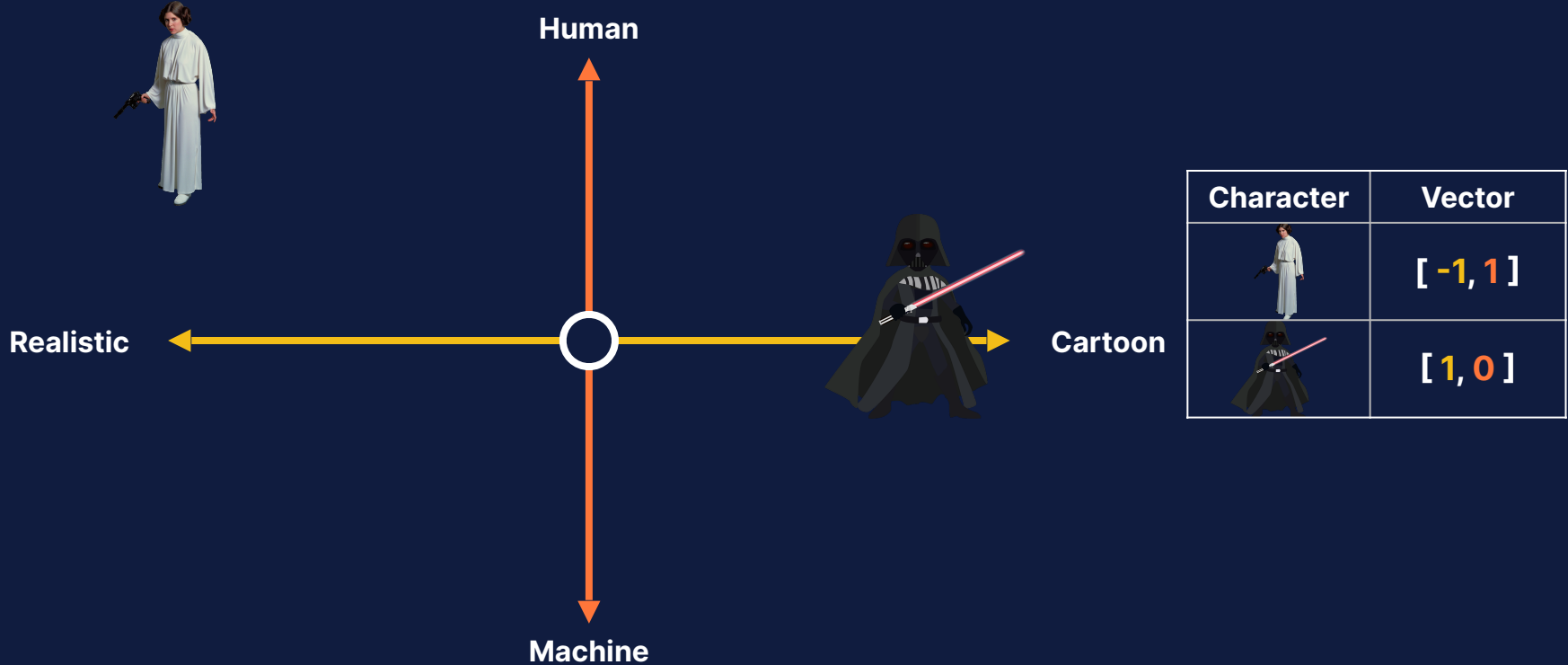
Embeddings represent your data

Example: 1-dimensional vector

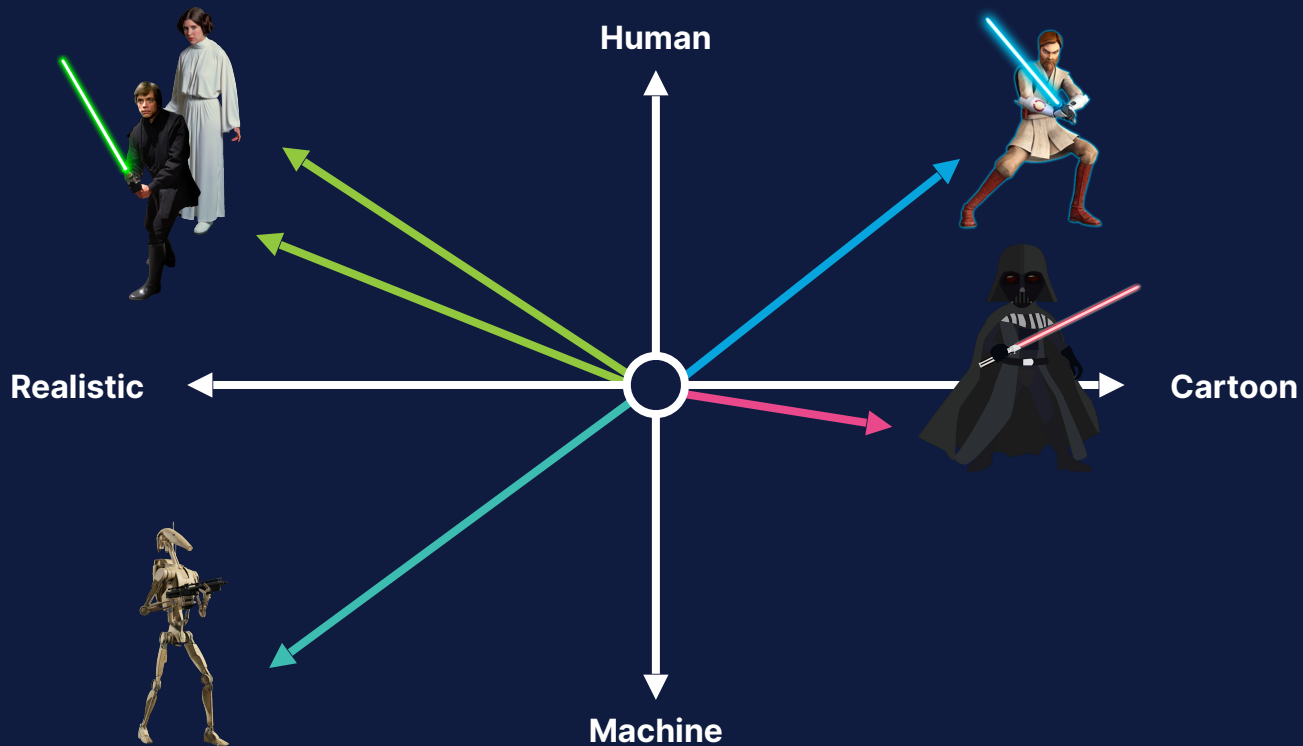


Character	Vector
	$[-1]$
	$[1]$

Multiple dimensions represent different data aspects

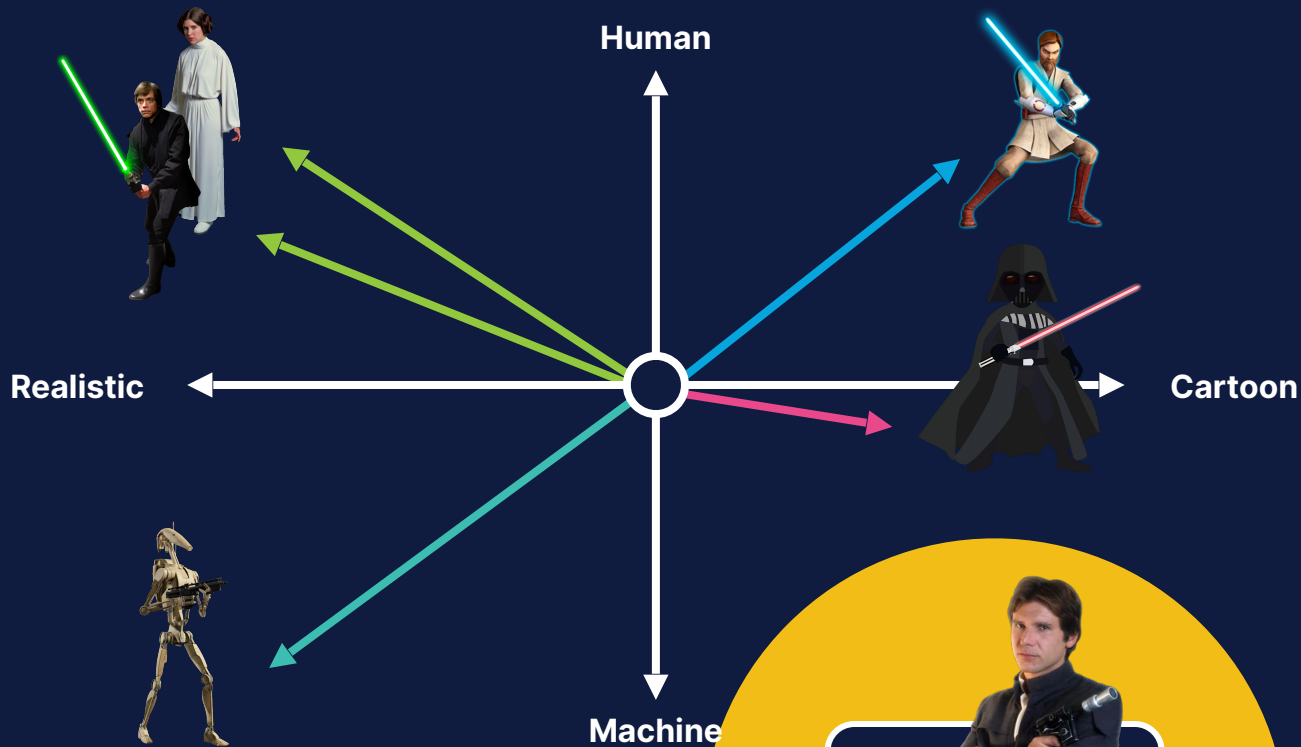


Similar data is grouped together

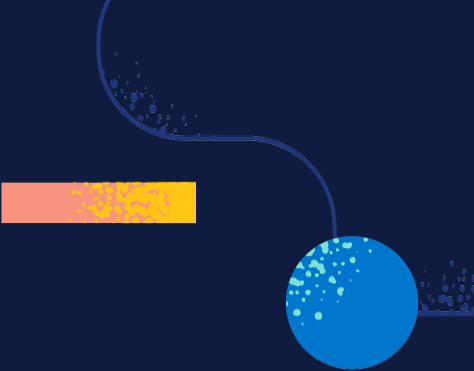


Character	Vector
	$[-1.0, 1.0]$
	$[1.0, 0.0]$
	$[-1.0, 0.8]$
	$[1.0, 1.0]$
	$[-1.0, -1.0]$

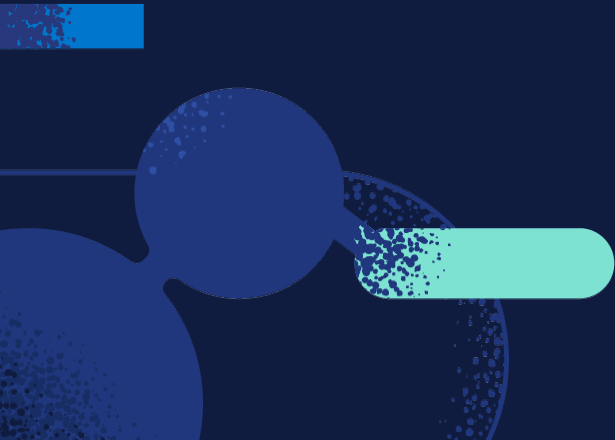
Vector search ranks objects by similarity (~relevance) to the query



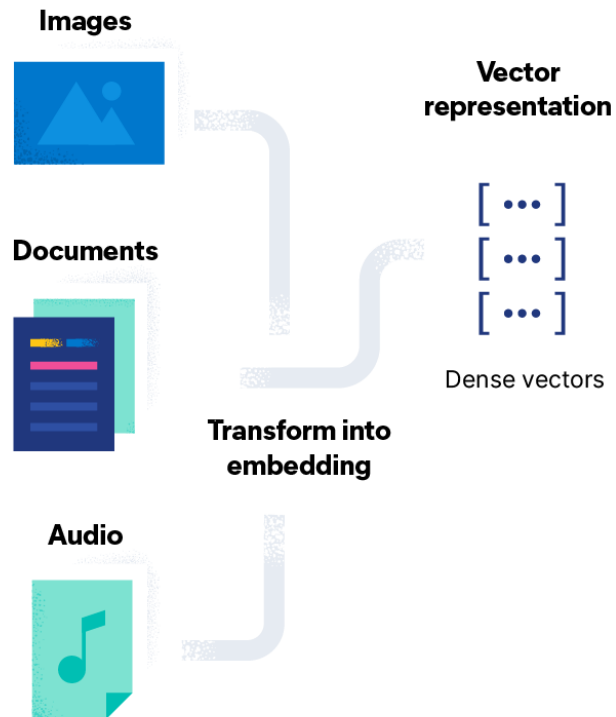
Rank	Result
Query	
1	
2	
3	
4	
5	



How do you index **vectors**?



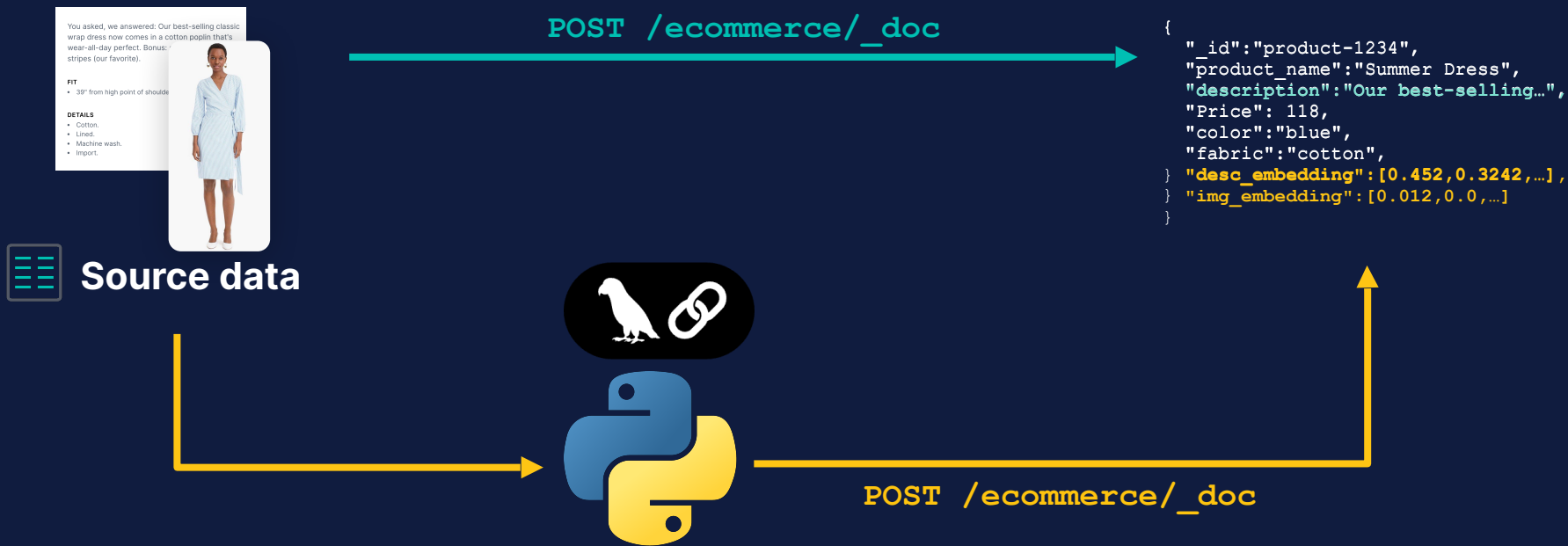
Architecture of Vector Search



dense_vector field type

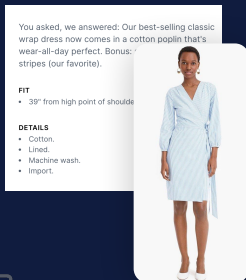
```
PUT ecommerce
{
  "mappings": {
    "properties": {
      "description": {
        "type": "text"
      }
      "desc_embedding": {
        "type": "dense_vector"
      }
    }
  }
}
```

Data Ingestion and Embedding Generation



With Elastic ML

commercial



Source data

```
{
  "_id": "product-1234",
  "product_name": "Summer Dress",
  "description": "Our best-selling classic wrap dress now comes in a cotton poplin that's wear-all-day perfect. Bonus: stripes (our favorite).",
  "Price": 118,
  "color": "blue",
  "fabric": "cotton",
}
```

POST /ecommerce/_doc



ML Inference pipelines [Add inference pipeline](#)

Inference pipelines will be run as processors from the Enterprise Search Ingest Pipeline

ml-inference-embedding-generation

[Actions](#)

Deployed pytorch text_embedding

ml-inference-emotional-analysis

[Actions](#)

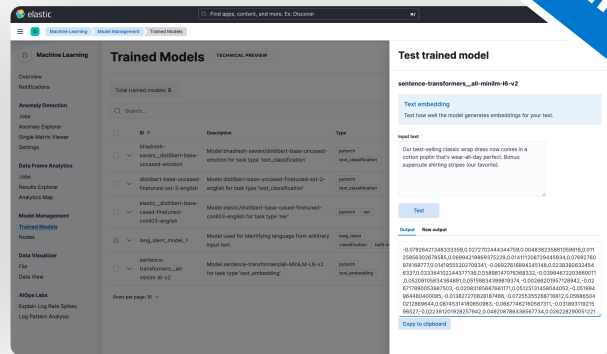
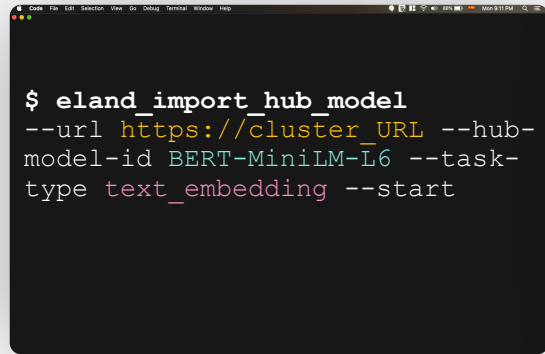
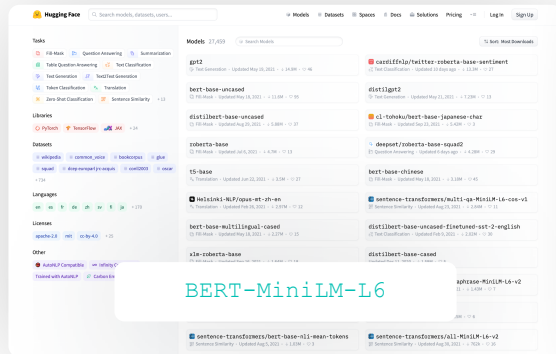
Deployed pytorch text_classification

[Learn more about deploying ML models in Elastic](#)

```
{
  "_id": "product-1234",
  "product_name": "Summer Dress",
  "description": "Our best-selling...",
  "Price": 118,
  "color": "blue",
  "fabric": "cotton",
  "desc_embedding": [0.452, 0.3242, ...]
}
```

Eland Imports PyTorch Models

Commercial



PyTorch



Select the appropriate model

Load it

Manage models



Elastic's range of supported NLP models

- **Fill mask model**

Mask some of the words in a sentence and predict words that replace masks

- **Named entity recognition model**

NLP method that extracts information from text

- **Text embedding model**

Represent individual words as numerical vectors in a predefined vector space

- **Text classification model**

Assign a set of predefined categories to open-ended text

- **Question answering model**

Model that can answer questions given some or no context

- **Zero-shot text classification model**

Model trained on a set of labeled examples, that is able to classify previously unseen examples

Third party fill-mask models

- BE
- Dis
- MP
- Ro

Third party text classification models

- BE
- De
- Dis
- Fi
- Tw

Third party named entity recognition models

- BE

Third party question answering models

- BE

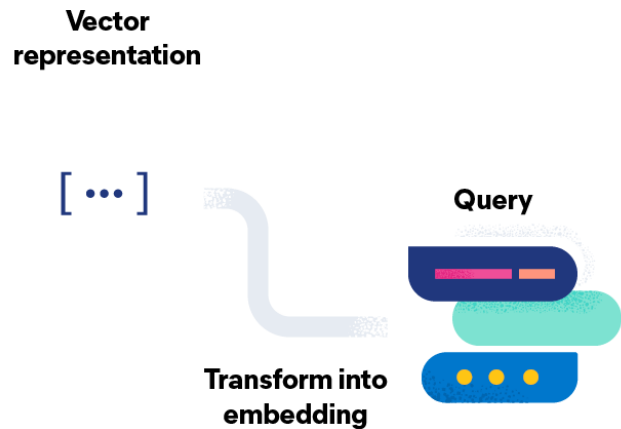
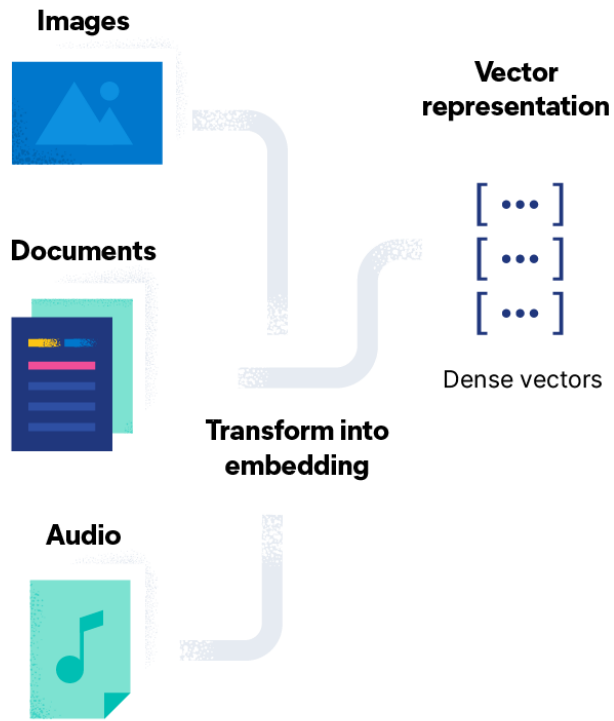
Third party text embedding models

Third party zero-shot text classification models

- BART large mnli
- DistilBERT base model (uncased)
- **DistilBart MNLI**
- MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices
- NLI DistilRoBERTa base
- NLI RoBERTa base
- SqueezeBERT

How do you search **vectors**?

Architecture of Vector Search



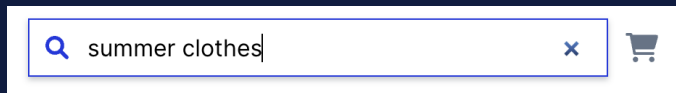
knn query

GET /ecommerce/_search

```
{
  "query" : {
    "bool" : {
      "must" : [{
        "knn" : {
          "field": "desc_embedding",
          "query_vector": [0.123, 0.244, ...]
        }
      }]
    },
    "filter" : {
      "term" : {
        "department": "women"
      }
    }
  },
  "size": 10
}
```

knn query (with Elastic ML)



Transformer model

```
GET /ecommerce/_search
{
  "query" : {
    "bool" : {
      "must" : [{
        "knn" : {
          "field" : "desc_embedding",
          "query_vector_builder" : {
            "text_embedding" : {
              "model_text" : "summer clothes",
              "model_id" : <text-embedding-model>
            }
          }
        }
      ]
    },
    "filter" : {
      "term" : {
        "department" : "women"
      }
    }
  },
  "size" : 10
}
```

semantic_text field type

new from 8.15

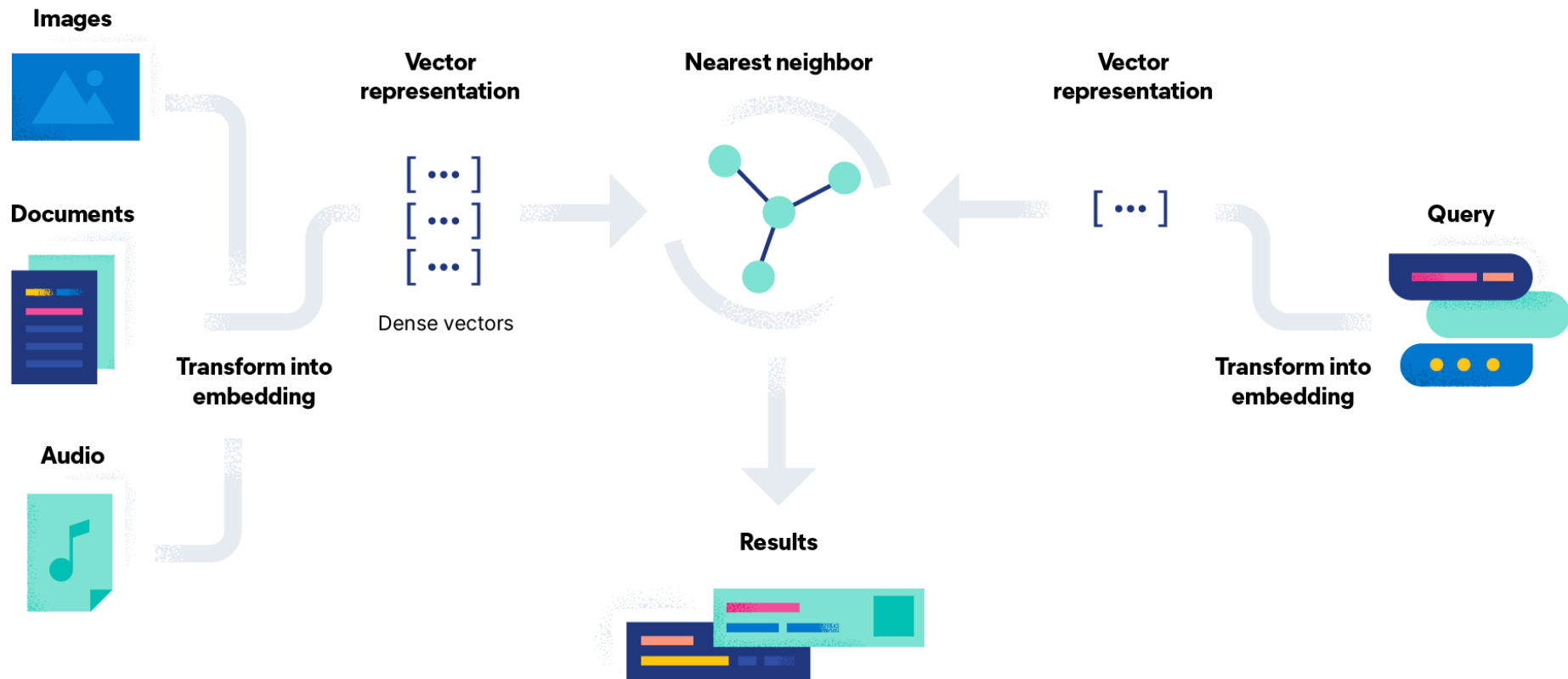
```
PUT /_inference/text_embedding/e5-small-multilingual
{
  "service": "elasticsearch",
  "service_settings": {
    "num_allocations": 1,
    "num_threads": 1,
    "model_id": ".multilingual-e5-small_linux-x86_64"
  }
}
```

```
PUT ecommerce
{
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "copy_to": [ "desc_embedding" ]
      }
      "desc_embedding": {
        "type": "semantic_text",
        "inference_id": "e5-small-multilingual"
      }
    }
  }
}
```

```
POST ecommerce/_doc
{
  "description": "Our best-selling..."
}
```

```
GET ecommerce/_search
{
  "query": {
    "semantic": {
      "field": "desc_embedding"
      "query": "I'm looking for a red dress for a DJ party"
    }
  }
}
```

Architecture of Vector Search



Choice of Embedding Model

Start with Off-the Shelf Models

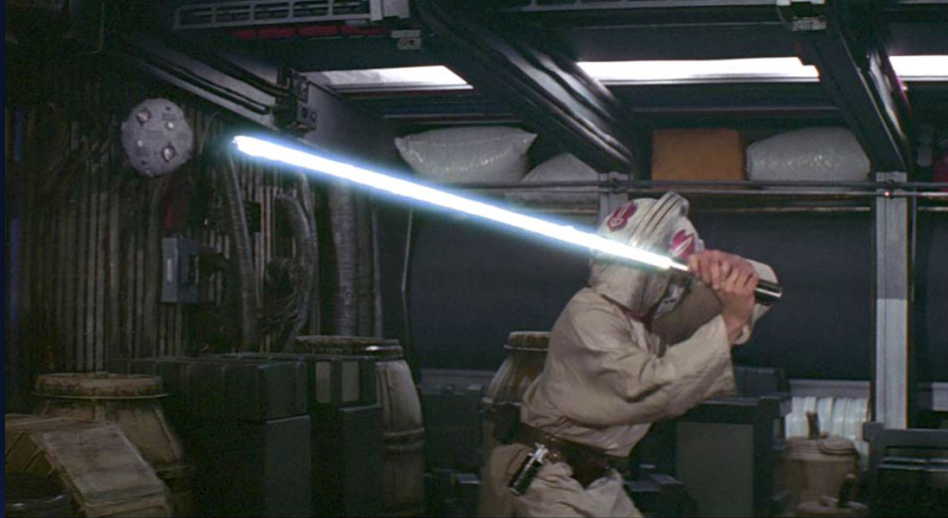
- Text data: Hugging Face (like Microsoft's E5)
- Images: OpenAI's CLIP

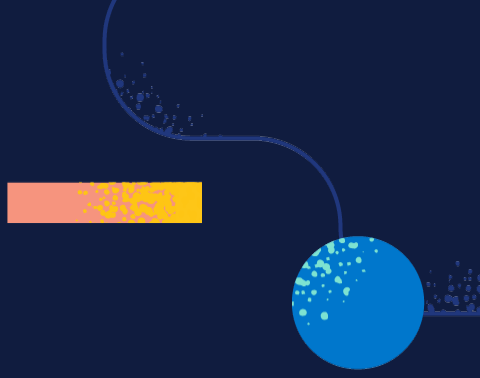
Extend to Higher Relevance

- Apply hybrid scoring
- Bring Your Own Model: requires expertise + labeled data

Problem

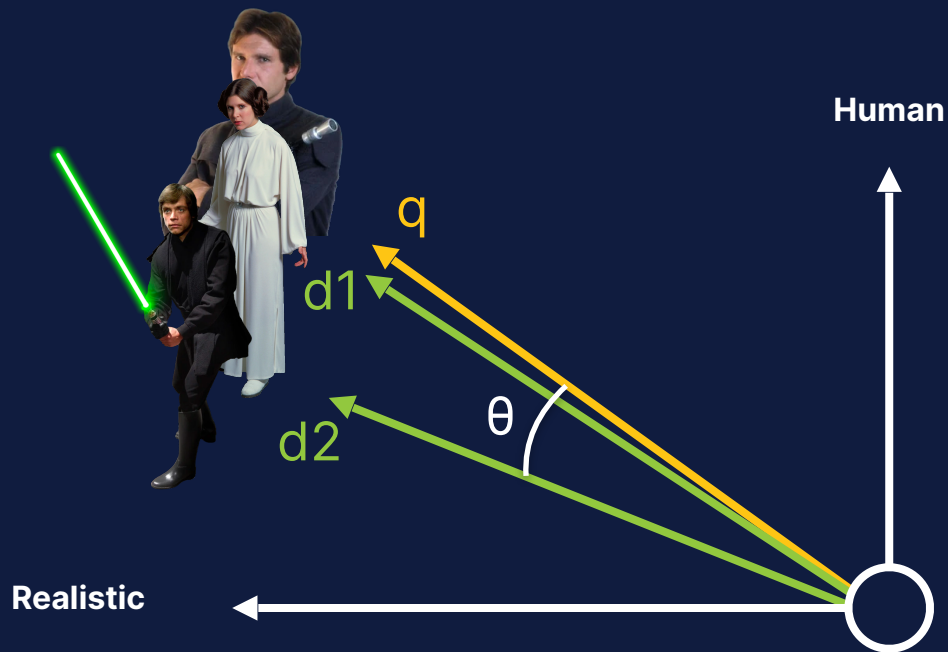
training vs actual use-case





But how does it really work?

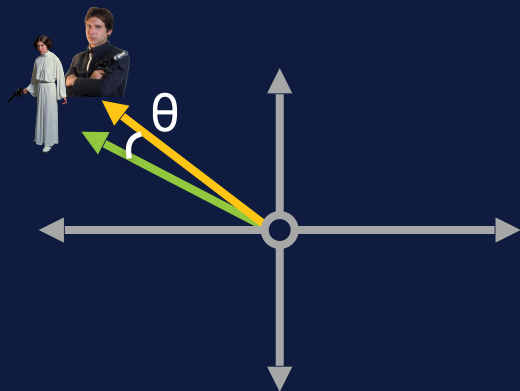
Similarity



$$\cos(\theta) = \frac{\vec{q} \times \vec{d}}{|\vec{q}| \times |\vec{d}|}$$

$$\text{_score} = \frac{1 + \cos(\theta)}{2}$$

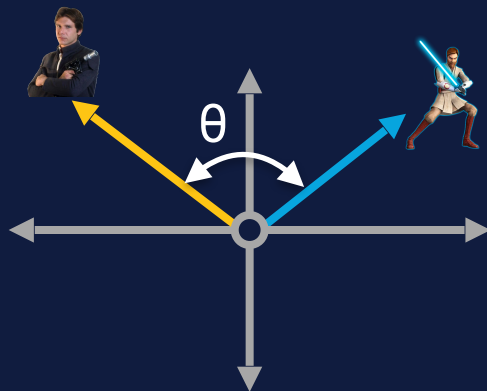
Similarity: cosine (cosine)



Similar vectors

θ close to 0
 $\cos(\theta)$ close to **1**

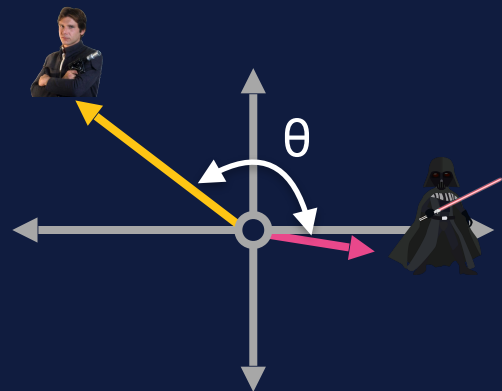
$$\text{_score} = \frac{1 + 1}{2} = 1$$



Orthogonal vectors

θ close to 90°
 $\cos(\theta)$ close to **0**

$$\text{_score} = \frac{1 + 0}{2} = 0.5$$



Opposite vectors

θ close to 180°
 $\cos(\theta)$ close to **-1**

$$\text{_score} = \frac{1 - 1}{2} = 0$$

Similarity: Dot Product (dot_product or max_inner_product)

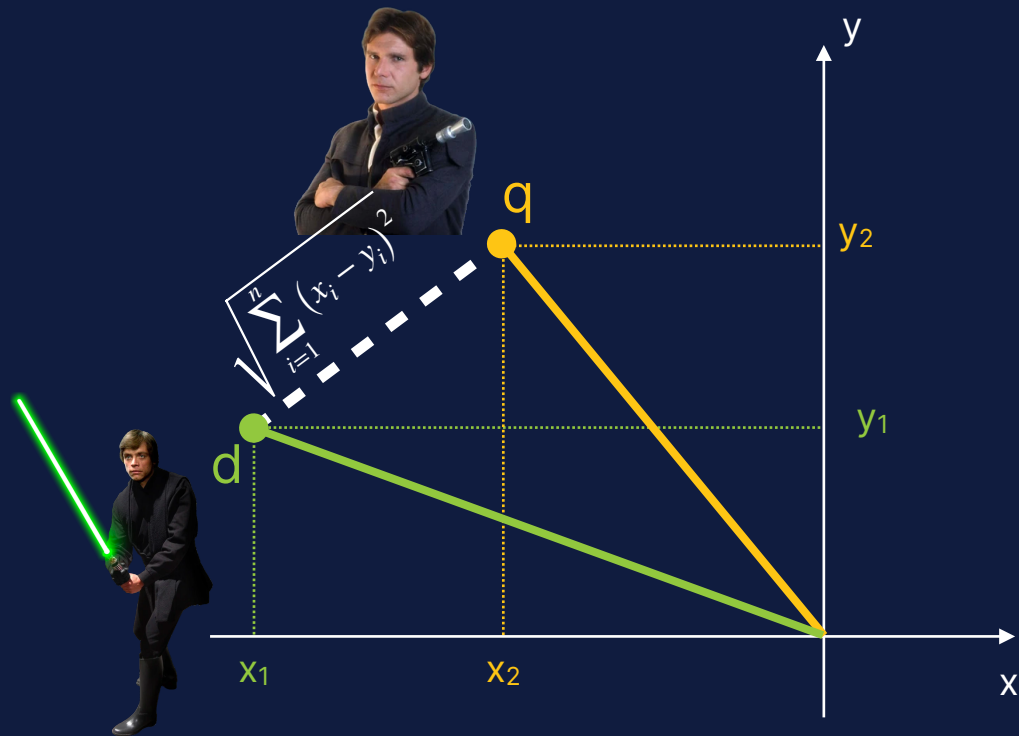


$$\vec{q} \times \vec{d} = |\vec{q}| \times \cos(\theta) \times |\vec{d}|$$

$$_{score}_{float} = \frac{1 + dot_product(q, d)}{2}$$

$$_{score}_{byte} = \frac{0.5 + dot_product(q, d)}{32768 \times dims}$$

Similarity: Euclidean distance (l2_norm)



$$l2_norm_{q,d} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$
$$_score = \frac{1}{1 + (l2_norm_{q,d})^2}$$

Brute Force



Hierarchical Navigable Small Worlds (HNSW)

One popular approach



HNSW: a layered approach that simplifies access to the nearest neighbor



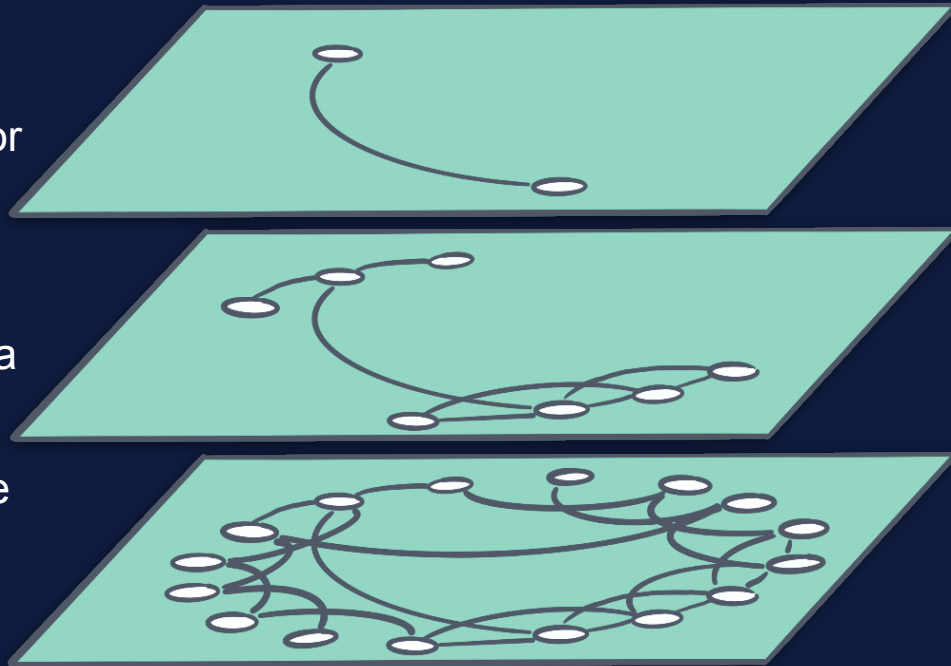
Tiered: from coarse to fine approximation over a few steps



Balance: Bartering a little accuracy for a lot of scalability



Speed: Excellent query latency on large scale indices



Scaling Vector Search

Vector search

1. Needs lots of memory
2. Indexing is slower
3. Merging is slow

* Continuous improvements in Lucene + Elasticsearch

Best practices

1. Avoid searches during indexing
2. Exclude vectors from `_source`
3. Reduce vector dimensionality
4. Use int8/int4/bit rather than float

Scalar Quantization

Elasticsearch
8.14+ default



float32

Recall: High
Precision: High
Rescore: Likely Not Needed

Full RAM Required



int8

Recall: Good
Precision: Good
Oversampling: Moderate

Rescore: Reasonable

4X RAM Savings



int4

Recall: Low
Precision: Low
Oversampling: Needed

Rescore: may be slower

8X RAM Savings



bit

Recall: Bad
Precision: Bad
Oversampling: Needed

Rescore: Expensive and Limiting

32X RAM Savings

BBQ aka Better Binary Quantization



float32



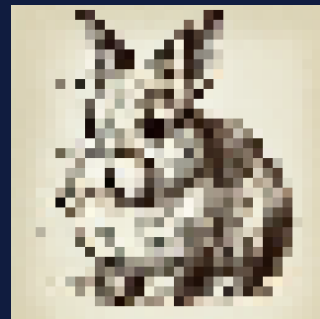
int8



int4



bit

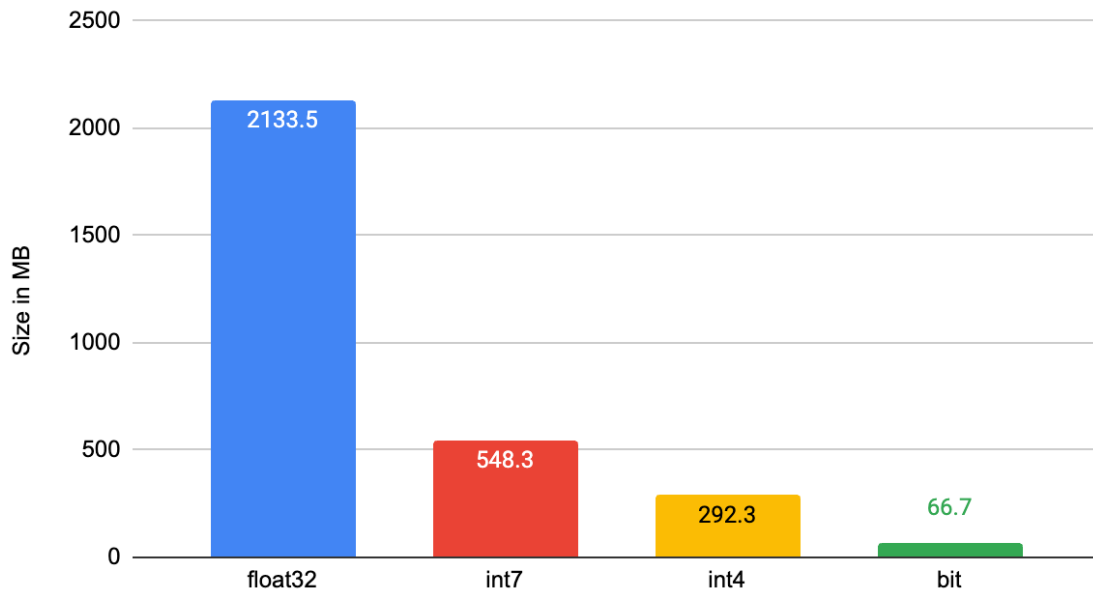


BBQ*

BBQ: 32X RAM savings.
Faster & more accurate than Product Quantization

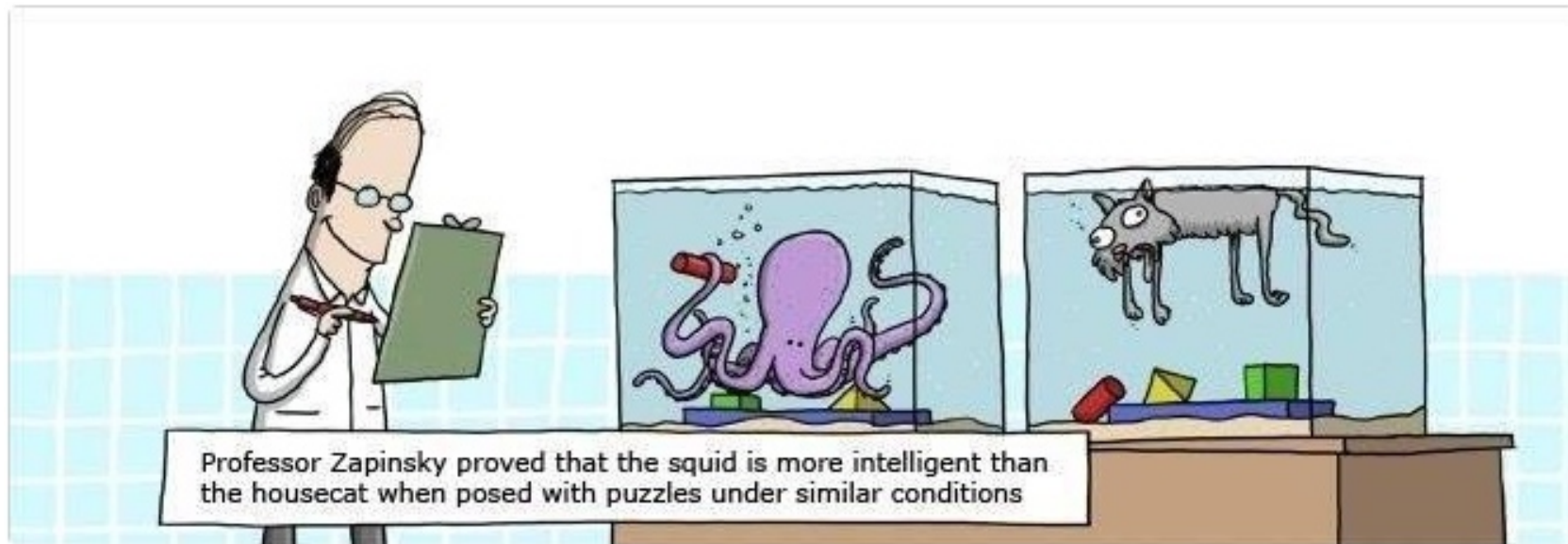
Memory required

Memory required in MB for 500k 1024 vectors



100M vectors?
Only 12GB!?! One single node.

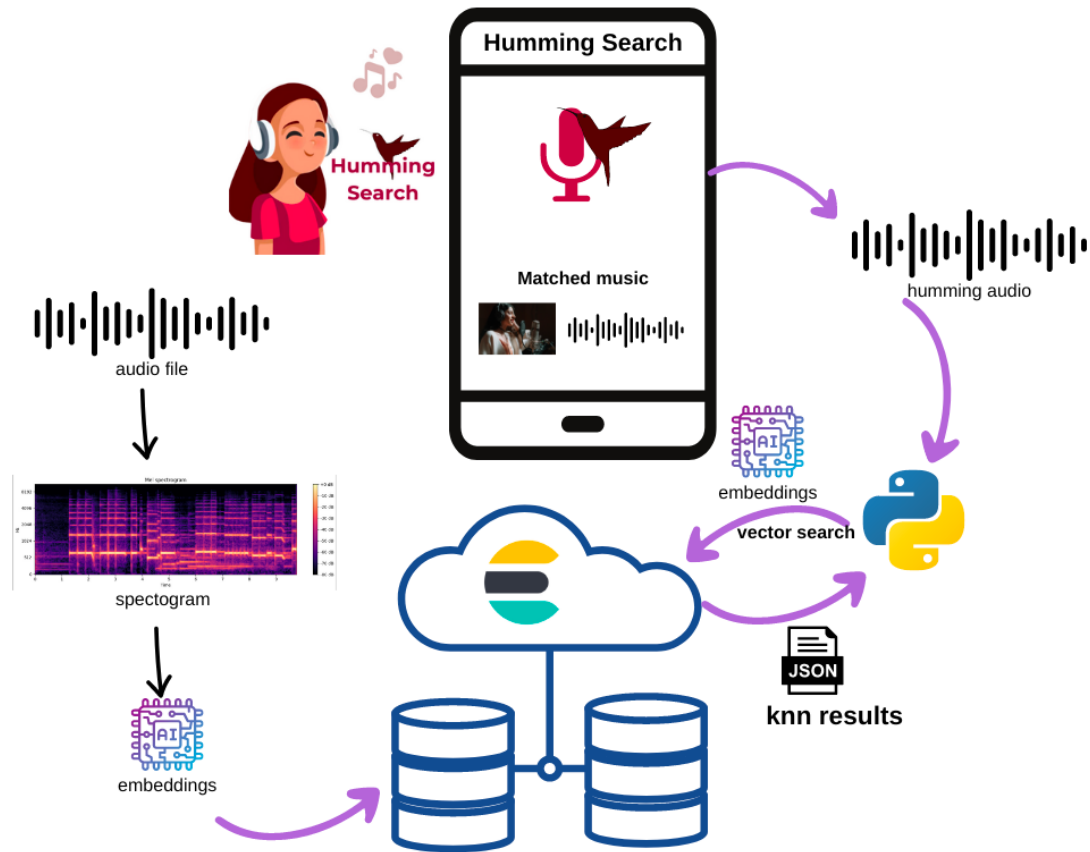
Benchmarking





<https://djdadoo.pilato.fr/>



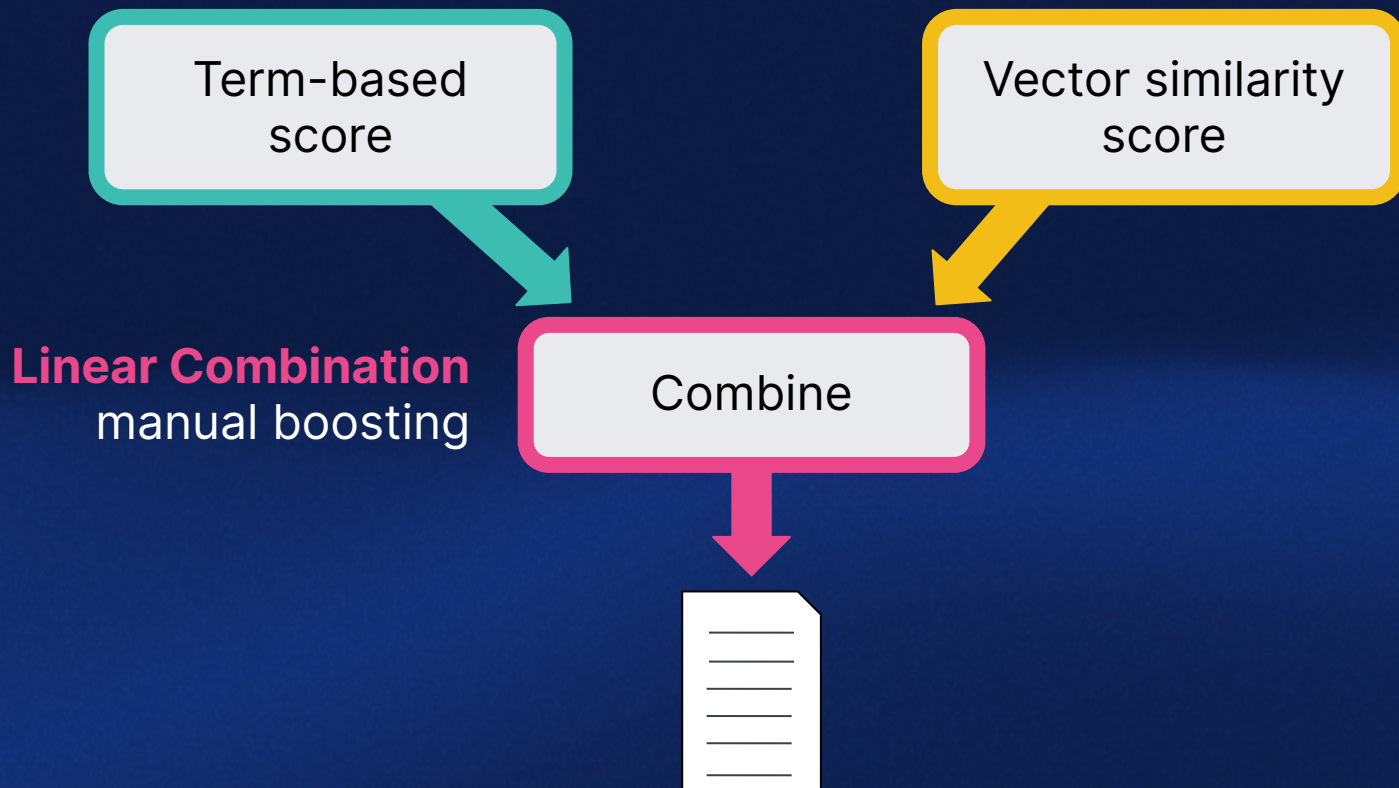


<https://github.com/dadoonet/music-search/>

Elasticsearch

You Know, for **Hybrid** Search

Hybrid scoring



```
GET ecommerce/_search
```

```
{
  "query" : {
    "bool" : {
      "must" : [{
        "match": {
          "description": {
            "query": "summer clothes",
            "boost": 0.1
          }
        }
      ]
    }, {
      "knn": {
        "field": "desc_embedding",
        "query_vector": [0.123, 0.244, ...],
        "boost": 2.0,
        "filter": {
          "term": {
            "department": "women"
          }
        }
      }
    }
  ],
  "filter" : {
    "range" : { "price": { "lte": 30 } }
  }
}
```

summer clothes

pre-filter

post-filter

```
PUT starwars
```

```
{  
  "mappings": {  
    "properties": {  
      "text.tokens": {  
        "type": "sparse_vector"  
      }  
    }  
  }  
}  
  
"These are not the droids you are looking for.",  
"Obi-Wan never told you what happened to your father."
```

```
GET starwars/_search
```

```
{  
  "query": {  
    "sparse_vector": {  
      "field": "text.tokens",  
      "query_vector": {  
        "lucas": 0.50047517,  
        "ship": 0.29860738,  
        "dragon": 0.5300422,  
        "quest": 0.5974301, ... }  
    }  
  }  
}
```

ELSER

Elastic Learned Sparse Encoder

sparse_vector

Not BM25 or (dense) vector

Sparse vector like BM25

Stored as inverted index

Commercial

Machine Learning Inference Pipelines

Inference pipelines will be run as processors from the Enterprise Search Ingest Pipeline

New Improve your results with ELSER ×

ELSER (Elastic Learned Sparse Encoder) is our **new trained machine learning model** designed to efficiently use context in natural language queries. This model delivers better results than BM25 without further training on your data.

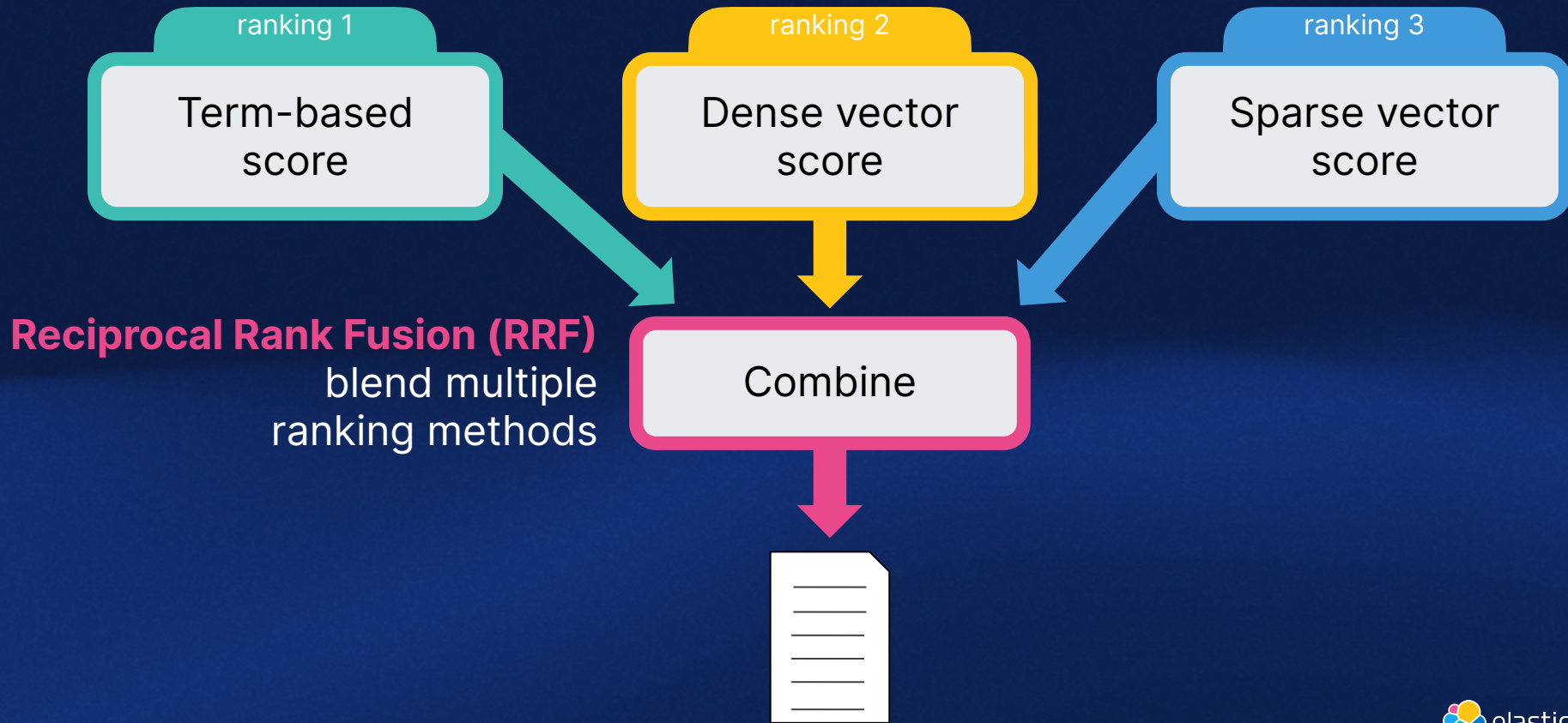
 Deploy

[Learn more](#) 

 Add Inference Pipeline

[Learn more about deploying Machine Learning models in Elastic](#) 

Hybrid ranking



Reciprocal Rank Fusion (RRF)

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k+r(d)}$$

D - set of docs

R - set of rankings as permutation on $1..|D|$

k - typically set to 60 by default

Dense Vector				BM25			
Doc	Score	r(d)	k+r(d)	Doc	Score	r(d)	k+r(d)
A	1	1	61	C	1,341	1	61
B	0.7	2	62	A	739	2	62
C	0.5	3	63	F	732	3	63
D	0.2	4	64	G	192	4	64
E	0.01	5	65	H	183	5	65



Doc	RRF Score
A	$1/61 + 1/62 = 0,0325$
C	$1/63 + 1/61 = 0,0323$
B	$1/62 = 0,0161$
F	$1/63 = 0,0159$
D	$1/64 = 0,0156$

```
GET index/_search
{
  "retriever": {
    "rrf": {
      "retrievers": [{
        "standard" { "query": {
          "match": {...}
        }
      }, {
        "standard" { "query": {
          "sparse_vector": {...}
        }
      }, {
        "knn": { ... }
      }
    ]
  }
}
```

Hybrid Ranking



BM25f

+

Sparse Vector

+

Dense Vector

ChatGPT

Elastic and LLM

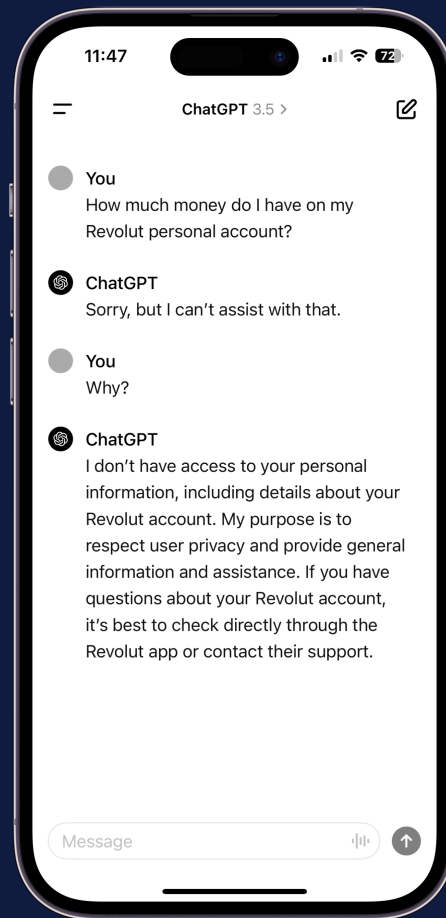
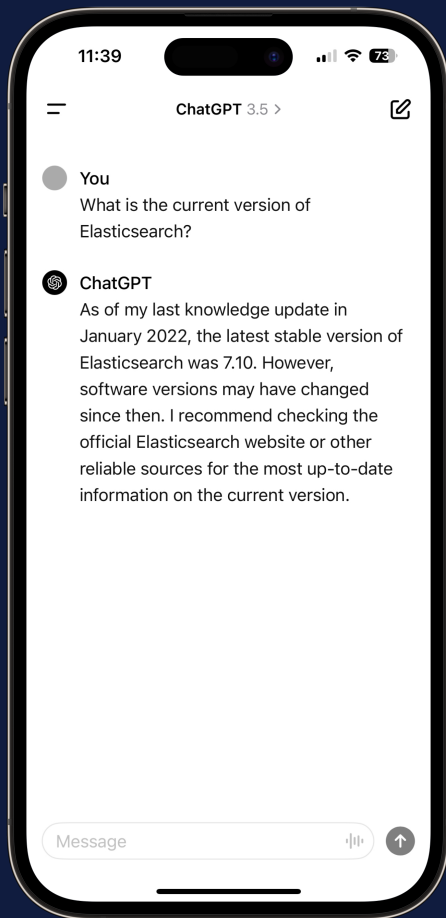
Gen AI

Search engines

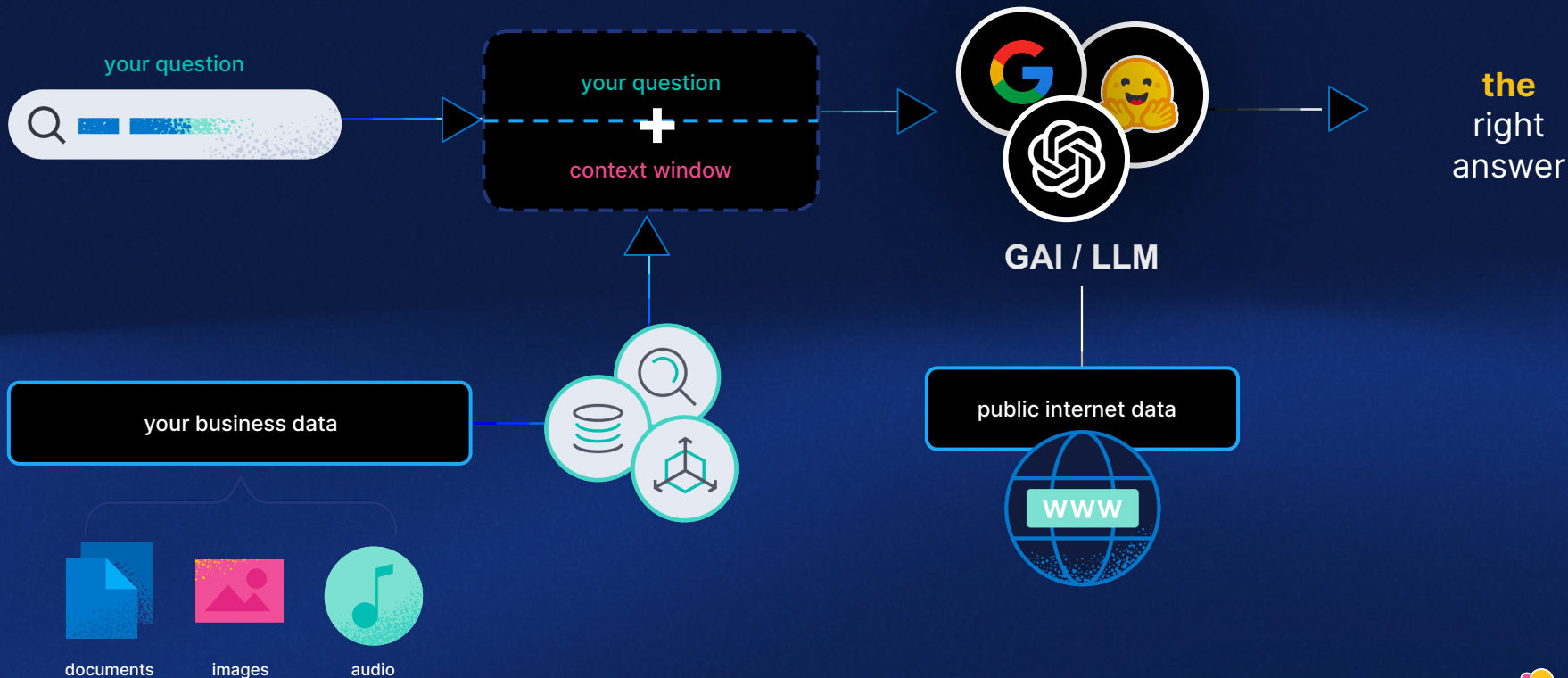


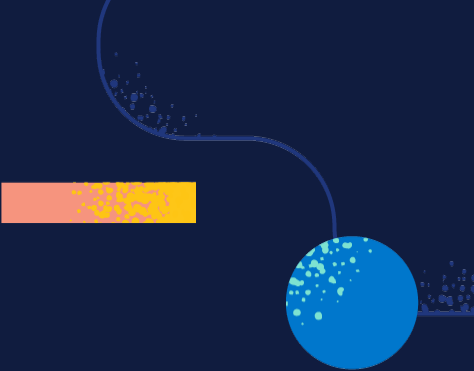
LLM: opportunities and limits





Retrieval Augmented Generation





Demo



Elastic Playground

Search your transactions:

This search is not enabled by Elastic and reflects the kind of functionality available to customers today.

[Submit](#)

Date	Account	Description	Value	Opening balance	Closing balance
18/06/24	EL03-130981-Transmission	Inbound payment made from EL03-130981-Transmission, St.james's Plac (STJ): 864dce1b-bb95-47d5-87dd-7d02f3b10c3f	7419.0	-825.0	6594.0
18/06/24	EL03-130981-Transmission	Purchase at merchant: Southeastern Grocers, LLC, location: Fayetteville,AR	82.0	6594.0	6512.0
18/06/24	EL03-130981-Transmission	Purchase at merchant: Müller Holding Ltd. & Co. KG, location: Glendale,AZ	188.0	6512.0	6324.0
17/06/24	EL03-130981-Transmission	Payment made from EL03-130981-Transmission to Elwood Erickson, Mitie Grp. (MTO): d37085fc-1382-4593-9cb8-26e5526bd9a0	533.0	20.0	-513.0
17/06/24	EL03-130981-Transmission	Payment made from EL03-130981-Transmission to Classie Johns, Barclays (BARC): 75b603a2-1c1b-45e9-a7ec-4a551bf98a8d	312.0	-513.0	-825.0
16/06/24	EL03-130981-Transmission	Purchase at merchant: E-MART Inc., location: Fayetteville,AR	31.0	51.0	20.0
14/06/24	EL03-130981-Transmission	Purchase at merchant: Dick's Sporting Goods, Inc., location: Montgomery,AL	182.0	329.0	147.0
14/06/24	EL03-130981-Transmission	Purchase at merchant: Valor Holdings Co., Ltd., location: Louisville,KY	96.0	147.0	51.0
13/06/24	EL03-130981-Transmission	Purchase at merchant: The Save Mart Companies, location:	34.0	363.0	329.0

Elasticsearch

You Know, for **Semantic** Search



Search a new era

David Pilato

 @dadoonet

 @pilato.fr

Normandie.ai

Les Journées Normandes des intelligences artificielles responsables

